PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04L 12/00, 12/26, G06F 13/00, 15/00, 3/00, 9/06, 17/30, H04M 3/42, H04B 1/00

A1

(11) International Publication Number: **WO 99/27680**

(43) International Publication Date: 3 June 1999 (03.06.99)

(21) International Application Number: PCT/US98/21239

(22) International Filing Date: 7 October 1998 (07.10.98)

(30) Priority Data:

60/066,227	20 November 1997 (20.11.97)	US
09/135,619	18 August 1998 (18.08.98)	US

(81) Designated States: AU, CA, CN, ID, JP, KR, MX, SG, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

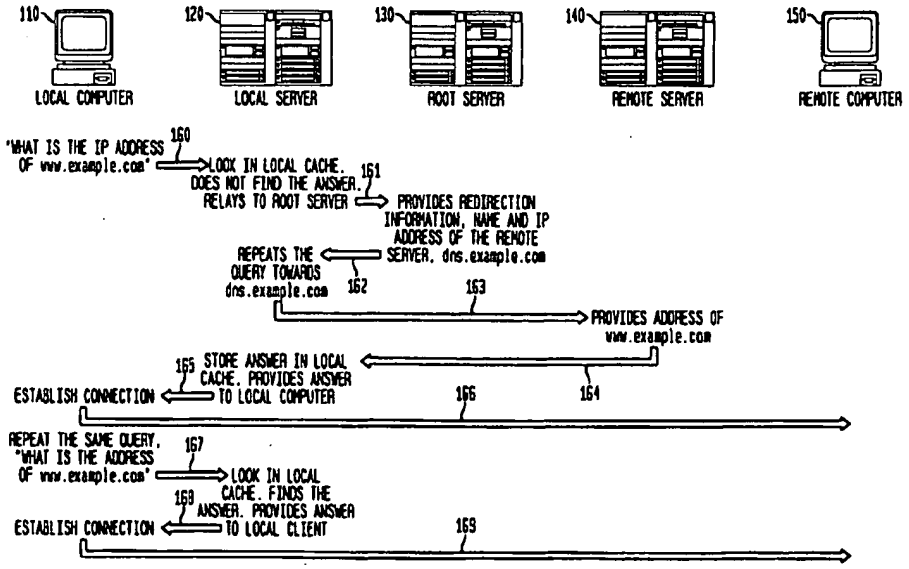
Published
With international search report.

(71) Applicant: BELL COMMUNICATIONS RESEARCH, INC. [US/US]; 445 South Street, Morristown, NJ 07960 (US).

(72) Inventor: HUITEMA, Christian; Apartment 119N, 77 Bleecker Street, New York, NY 10012 (US).

(74) Agents: GIORDANO, Joseph et al.; Bell Communications Research, Inc., c/o International Coordinator, Room 1G112R, 445 South Street, Morristown, NJ 07960 (US).

(54) Title: ENHANCED DOMAIN NAME SERVICE



(57) Abstract

A system (300) prefetches most frequently used domain names and stores the domain name data (320) at local cache servers (310). It generates validity codes (330) to enable error checking for valid domain names at the local cache servers (310) without accessing root servers (130). A cache server (310) obtains, stores, and propagates updated or new DNS data to local cache servers (340) at predetermined intervals. Users can obtain Internet protocol addresses of domain names directly from local cache servers (310), thus eliminating processing delays over the Internet.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Description

Enhanced Domain Name Service

Related Applications

This application is based on and claims the priority of provisional application, Serial No. 60/066,227 filed on November 20, 1997, the contents of which are hereby incorporated by reference.

Background Art

The present invention relates generally to enhanced domain name servers (DNS), and more particularly, to efficiently processing domain name (DN) queries in a network.

The Internet has experienced explosive growth in recent years and provides a vast resource of information on a wide array of topics. The Internet requires DNS for providing Internet Protocol (IP) addresses corresponding to the DNs. The response time of any DNS query, however, is highly dependent on the latency of the Internet. In addition, the availability of any response varies with the packet loss rate of the Internet. DNS developers attempted to address these problems by including a cache memory in local servers. Fig. 1 is a state diagram showing a process flow of traditional DNS processing.

To access a remote computer over the Internet, a user types in the DN or URL of that remote computer (*e.g.*, www.example.com) using a browser at a local computer 110 (step 160). Most Internet connections begin by a query to a local server 120 to obtain the IP address of that computer (*e.g.*, 128.96.41.1). To do so, local server 120 looks in its local cache storing previous queries. If the current query is for an IP address that matches a previous query, local server 120 can respond to the query immediately using the information stored in the local cache.

If local server 120 does not find the answer in the local cache, however, local server 120 relays the query to a root server 130 (step 161). Root server 130, or more precisely the "top level domain server," knows the address of all "remote servers" that manage second level names (*e.g.*, example.com). Root server 130 transmits redirection information, *i.e.*, name and IP address of the remote server (*e.g.*, dns.example.com) to local server 120 (step 162). Thereafter, local server 120 repeats the query, this time to the specified remote

server 140 (*e.g.*, dns.example.com) (step 163). Remote server 140, in turn, transmits the IP address of the requested DN to local server 120 (step 164). Local server 120 stores a copy of this answer in its local cache and provides an answer to local computer 110 (step 165). Using the IP address, local computer 110 establishes connection with remote computer 150 (step 166).

Subsequently, if local computer 110 repeats the same query (step 167), local server 120 looks in the local cache and will find the IP address. Local server 120 then transmits the address to local computer 110 without having to access other components of the network (step 168), thus enabling local computer 110 to establish connection to remote computer 150 (step 169).

Referring to Fig. 2, a special case occurs when a user mistypes or guesses the DN, making up such names as "www.i-dont-really-know.com" (step 200). Similar to the above-explained process flow, local server 120 checks its local cache. But since local server 120 will not find the DN in the local cache, it relays the query to root server 130 (step 201). Thereafter, root server 130 checks a reference database to determine whether the DN exits and returns an error message to local server 120 (step 202). Local server 120, in turn, relays the error message to local computer 110 (step 203).

Some local servers keep a copy of these error messages, but typically only for a short period of time since a name that is not found one day may well be registered the next. Moreover, copies of error messages are only useful if users keep repeating the exact same mistake, an unlikely event. In practice, most wrong guesses result in a transaction to one of the root servers, thus extending the DN processing time and subjecting communications to problems with Internet traffic.

Moreover, due to the dramatic increase in the size of the Internet since the late 1980's, the caching technique no longer provides high quality results. Typically, fewer than 85% of queries are served in less than three seconds. This is expected to worsen as the Internet grows.

Therefore, it is desirable to increase the efficiency in processing DN queries for connecting local computers to remote computers.

It is also desirable to provide efficient processing of invalid DNs.

Disclosure of the Invention

Accordingly, the present invention is directed to an enhanced DN service that substantially obviates one or more of the problems due to limitations and disadvantages of the related art.

Specifically, a method consistent with the present invention for processing DN requests comprises the following steps. Initially, the system prefetches most frequently used (MFU) domain names and associated domain addresses. It then transmits to a local server the prefetched MFU DNs and the associated domain addresses, wherein the local server uses the prefetched MFU DNs and the associated domain addresses to process DN requests.

A system consistent with the present invention for processing DN requests comprises prefetching means and transmitting means. Initially, the prefetching means prefetches MFU DNs and associated domain addresses. The transmitting means then transmits to a local server the prefetched MFU DNs and the associated domain addresses, wherein the local server uses the prefetched MFU DNs and the associated domain addresses to process DN requests.

Brief Description of the Drawings

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate the invention and together with the description, serve to explain the principles of the invention.

In the drawings,

Fig. 1 is a state diagram showing a process flow of traditional DNS processing of a DN request;

Fig. 2 is a state diagram showing a process flow of traditional DNS processing of a DN request for an invalid DN;

Fig. 3 is a diagram showing an example of a network consistent with one embodiment of the present invention;

Fig. 4 is a functional block diagram showing service elements of the network consistent with one embodiment of the present invention;

Fig. 5 is a state diagram showing a process flow of DNS processing consistent with one embodiment of the present invention; and

Fig. 6 is a state diagram showing a process flow of DNS processing of an invalid DN consistent with one embodiment of the present invention.

Best Mode for Carrying Out the Invention

Reference will now be made in detail to the present preferred embodiment of the invention, an example of which is illustrated in the accompanying drawings. Where appropriate, the same reference numerals refer to the same or similar elements. The appended claims define the scope of the invention; the following description does not limit that scope.

Fig. 3 shows an example of a network environment consistent with one embodiment of the present invention. In network 300, local computer 110 is coupled to a local cache server 310 over network connections. Local cache server 310 also connects to a cache server 340 and root server 130. Local cache server 310 includes a most frequently used domain names (MFU DNs) table 320 and a validity code table 330. Local cache server 310 preferably is a high-end network server having capacity to hold a few million records in its database and high-speed network access.

Fig. 4 is a functional block diagram showing service elements of network 300 consistent with one embodiment of the present invention. A central DNS cache server includes a central DNS database 410, a spider lookup 415, a refresh process 420, a cache administrator 425, a feedback process 430, ISP control cache servers 435, replicators 440, and edge cache servers 445. In one embodiment consistent with the present invention, the central DNS cache server can be operated by a commercial service, which would send updates to ISP control cache servers 435, typically operated by Internet Service Providers (ISPs). ISP control cache server 435, in turn, replicates the cache updates to edge cache servers 445. The updates are preferably sent over TCP-IP connections. ISP control cache servers 435 or edge cache servers 445 of Fig. 4 are examples of local cache server 310 in Fig. 3.

According to the present invention, local cache servers 310, located near local computers 110, contain a large portion of the DNS data. Thus, in most cases, requests from local computers 110 will be serviced directly by local cache servers 310. The DNS data contained in local cache servers 310 are not collected by the local cache servers 310 themselves, as in the prior art systems, but rather are provided by cache servers 340. Specifically, cache servers 340 update local cache servers 310 on a predetermined basis by providing DNS records consisting preferably of three data sets. The type and scope of data included in the MFU DNs may vary, however.

The first data set is a list of the most active names together with the corresponding name and address records, *i.e.*, DNS records of type NS (DNS record containing DNS names of the DNS servers that hold information about the specified domain) and type A (DNS record containing the IP address of a computer identified by a DNS name). For each DNS record, central DNS database 410 stores the name (name of the node to which the resource record pertains), type and class of the record, time to live (TTL) variable, length (specifying the length of the data field), and data (field describing the resource). TTL is the estimated life expectancy during which the DNS record remains current. The TTL can be set and reset to a certain predetermined value including periodic or discrete preset times.

The second data set is a complete listing of the existing names at the root level and, preferably, also second level DNs of the most frequently accessed domain, *e.g.*, ".com." A listing of second level domains is a flat file of names such as "bellcore.com." These names have two components, the second level domain name and the name of the "top level domain." This data set contains a full listing of all names that are registered in a monitored top level domain. The present invention generates validity codes using a standard technique, *e.g.*, a hash coding (explained below), and produces for each DN a hash code, which is preferably an integer number. A complete list of hash codes or validity codes, but not the list of names, is transmitted to local cache servers 310 for use in error checking as will be explained below.

The third data set is a listing of the highest levels of the DNS inverse tree used by servers to locate the origin of Internet connections. The inverse tree contains names that are derived from IP addresses. For example, the address "192.4.18.101" can be used to build the inverse name "101.18.4.192.in-addr.arpa." These names are used by servers to retrieve the domain names of clients. Each level of the naming hierarchy may be handled by a different server. The first level is typically handled by a registry, the second level by an Internet Service Provider or by a large organization, and the third level by a retail ISP or by the manager of a local network, *i.e.*, "edge cache servers." By caching DNS information at the first two levels, the queries can be directed immediately to the remote server that can transform the inverse name into the actual name of the machine that uses the IP address, in the present example, "seawind.bellcore.com."

Web lookup 415 employs web spiders, which are software processes that visit web sites and automatically explore their contents, to obtain MFU DNs. From the web content, web spiders extract names of other sites, which the web spiders can visit next, thus, exploring a large part of the Internet. Web spiders are typically used to build large index databases for index servers such as Yahoo, Altavista, and Excite. One embodiment consistent with the present invention employs web spiders to discover new web sites. Web spiders only obtain an approximation of the MFU DNs because they collect most frequently accessed references, which is a different property from MFU DNs. The collection of MFU DNs obtained by web spiders are supplemented by feedback process 430. Feedback process of ISP control cache servers 435 and edge local cache servers 445 collect statistics of the MFU DNs and send this data to central DNS database 410.

Refresh process 420 queries central DNS database 410 at intervals corresponding to TTL to determine whether the information in central DNS database 410 has changed to ensure that local cache server 310 always contains up-to-date information. In one embodiment consistent with the present invention, refresh process 420 need not conduct a full update. In many cases, TTL may be set to a relatively short value, such as one day, because managers of the records might update the next day. DNS database 410, in such cases, may be refreshed every day. In one preferred embodiment of the present invention, TTL is set to a level so that the information would not be updated too frequently, which would effectively preclude caching.

There is no need to perform a full update unless the information actually changes. If the information does not change, cache administrator 425 only needs to send a message that essentially revalidates the current information. If refresh process 420 determines that the information actually changed, cache administrator 425 extracts the information from central DNS database 410 and selects only actual updates or new information to propagate to ISP control cache servers 435 and edge cache servers 445. For example, an update is necessary when a new DN is added to central DNS database 410 by spider lookup 415 or feedback process 430. Replicator 440 of ISP control cache server 435 replicates the updates or new information to the next level of cache servers, *i.e.*, edge cache servers 445. Replicator 440 of edge cache servers 445, in turn, replicates the updates or new information to other cache servers.

Thus, referring to Fig. 5, according to one embodiment consistent with the present invention, cache server 340 transmits DNS data to local cache server 310 at predetermined intervals, or TTL (step 501). The data includes the three types of data sets explained above, and subsequent transmissions need only contain updates to the existing DNS data at local cache server 310. Local cache server 310 receives the DNS data or updates and updates MFU DNs table 320 (step 502). MFU DNs table 320, containing the three data sets, is preferably stored in cache for fast access.

When a user types in a user request, *i.e.*, the DN or URL of a remote computer (*e.g.*, www.example.com) using a browser at a local computer 110 (step 502), local cache server 310 accesses MFU DNs table 320. If local cache server 310 finds the answer, which will be the case in most instances, local cache server 310 transmits the corresponding IP address to local computer 110 (step 503). Using the IP address, local computer 110 establishes connection with remote computer 150 (step 504). Thus, local cache server 310 responds to queries directly, significantly reducing DNS processing time.

A different scenario occurs for DN requests of invalid names. In the present invention, cache server 340 generates validity codes of valid DN names to be stored in validity code table 330 of local cache server 310. In this manner, local cache server 310 checks its local cache to determine whether the requested DN is valid without having to access root server 130. Since the memory capacity of local cache at local cache server 310 is limited, in one embodiment consistent with the present invention, local cache server 310 stores the validity codes of all top level domains as well as second level DNs of the most frequently accessed domain, *e.g.*, ".com."

Accordingly, referring to Fig. 6, root server 130 of ".com" provides a full listing of second level DNs to cache server 340 (step 600). Cache server 340 preferably generates validity codes of the top level domains and ".com" using hash coding. Cache server 340 transmits the validity codes to local cache server 310 (step 601). Although there are many ways to implement the hash coding, the following Java method provides one example:

```
int hash (String domainName)
{
        int x, r, l;
        char c;

        x = 0;
        for (l=0; l<domainName.length(); l++){
                r = (x>>24)&255;
                x <<= 8;
                x |= r;
                x ^= (r<<1);
                c = domainName.charAt(l);
                if (c == '.')
                        x ^= 37;
                else if (c == '-')
                        x ^= 36;
                else
                        x ^= ((Character.digit(c, 36))&255);
        }
        return(x);
}
```

This algorithm returns a 32 bit integer, whose value is a function of the domain name:

Domain name	Hash code
bellcore.com	370308165
inria.fr	1008476717
nsa	622275647
nsa.gov	623459329

Other hash coding techniques may be implemented to generate codes requiring much less memory than the full DNs to reduce the memory requirement of local cache server 310. The same technique should be used consistently, however, within a single network platform by various components, *e.g.*, local cache server 310, cache server 340, and cache administrator 425. Additionally, the technique implemented should preferably be programmed to return results that only depend on the DNs and not the supporting platform.

Accordingly, when a user types in the DN or URL of a remote computer (*e.g.*, www.i-dont-really-know.com) using a browser at a local computer 110 (step 602), local cache server 310 accesses MFU DNs table 320. If local cache server 310 does not find the answer, rather than relaying the query to root server 130, local cache server 310 generates a check code of the DN in the query using hash coding or other technique implemented in network 300. If the generated check code is not in validity code table 330, local cache server 310 transmits an error message to local computer 110 (step 603). Thus, local cache server 310 reduces processing time for queries for invalid DNs by eliminating the need to access root server 130. In the rare instance when the DN is not in the MFU DNs table 320 and the check code matches one of the entries in validity code table 330, local cache server 310 processes the DN query in the traditional manner.

Systems and methods consistent with the present invention improve the efficiency of DN processing by storing MFU DNs and validity codes at local cache servers. Such systems and methods process DN queries without accessing remote servers or root servers, thus, significantly reducing processing time. The central server is periodically updated with new and revised DN data and the central server propagates the updates to edge cache servers.

It will be apparent to those skilled in the art that various modifications and variations can be made in the systems and methods of the present invention without departing from the scope or spirit of the invention. For example, Internet has been used as an exemplary network setting. The teachings of the present application, however, may be easily adapted and applied in other network settings by one ordinary skilled in the art. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and examples be considered as exemplary only, with the true scope and spirit of the invention indicated by the following claims.

Claims

1. A method of processing domain name requests at a local computer, the local computer connected to a local cache server storing prefetched most frequently used (MFU) domain names and associated domain addresses, the method comprising the steps of:

receiving a user request for a domain name;

transmitting a request to the local cache server for a domain address corresponding to the received domain name; and

receiving from the local cache server the domain address corresponding to the received domain name for connection with a remote computer having the received domain address.

2. The method of claim 1, wherein the prefetched MFU domain names and the associated addresses are prefetched by a web spider.

3. The method of claim 1, wherein the prefetched MFU domain names and the associated addresses are prefetched from other local cache servers in the network.

4. A method of supporting a local cache server of a network for processing domain name requests, the method comprising the steps of:

prefetching most frequently used (MFU) domain names and associated domain addresses; and

transmitting to the local cache server the prefetched MFU domain names and the associated domain addresses, wherein the local cache server uses the prefetched MFU domain names and the associated domain addresses to process domain name requests.

5. The method of claim 4, wherein the prefetching step includes the substep of

obtaining the MFU domain names and the associated addresses using a web spider.

6. The method of claim 4, wherein the prefetching step includes the substep of

obtaining the MFU domain names and the associated addresses from local cache servers in the network.

7. The method of claim 4, wherein the prefetching step includes the substep of

prefetching the MFU domain names and the associated addresses on a predetermined basis.

8. A method of supporting error functions of a local cache server of a network for processing domain name requests, the method comprising the steps of:

collecting a plurality of valid domain names;

generating validity codes corresponding to the plurality of valid domain names; and

transmitting to the local cache server the validity codes.

9. The method of claim 8, wherein the generating step includes the substep of

generating the validity codes using a hash coding.

10. A method of processing domain name requests at a local cache server, the local cache server connected to a local computer of a network, the method comprising the steps of:

storing in a memory prefetched most frequently used (MFU) domain names and associated domain addresses;

receiving a request from the local computer for a domain address corresponding to a domain name;

determining whether the received domain name matches any of the prefetched MFU domain names; and

transmitting to the local computer the domain address associated with the received domain name.

11. The method of claim 10, further including the step of

determining the validity of the received domain name.

12. The method of claim 11, further including the steps of

storing in the memory validity codes for valid domain names,

generating, in response to a determination that the received domain name does not match any of the prefetched MFU domain names, a check code of the received domain name, and

determining whether the check code matches any of the validity codes.

13. The method of claim 12, further including the step of

transmitting to the local computer, in response to a determination that the check code matches one of the validity codes, a signal indicating that the received domain name is valid.

14. The method of claim 12, further including the step of

transmitting to the local computer, in response to a determination that the check code does not match any of the validity codes, a signal indicating that the received domain name is invalid.

15. The method of claim 10, further including the step of

receiving the prefetched MFU domain names and the associated domain addresses from a cache server.

16. A local computer for processing domain name requests, the local computer connected to a local cache server storing prefetched most frequently used (MFU) domain names and associated domain addresses, the local computer comprising:

means for receiving a user request for a domain name;

means for transmitting a request to the local cache server for a domain address corresponding to the received domain name; and

means for receiving from the local cache server the domain address corresponding to the received domain name for connection with a remote computer having the received domain address.

17. The local computer of claim 16, wherein the prefetched MFU domain names and the associated addresses are prefetched by a web spider.

18. The local computer of claim 16, wherein the prefetched MFU domain names and the associated addresses are prefetched from other local cache servers in the network.

19. A cache server for supporting a local cache server of a network for processing domain name requests, the cache server comprising:

means for prefetching most frequently used (MFU) domain names and associated domain addresses; and

means for transmitting to the local cache server the prefetched MFU domain names and the associated domain addresses, wherein the local cache server uses the prefetched MFU domain names and the associated domain addresses to process domain name requests.

20. The cache server of claim 19, wherein the prefetching means includes

means for obtaining the MFU domain names and the associated addresses using a web spider.

21. The cache server of claim 19, wherein the prefetching means includes

means for obtaining the MFU domain names and the associated addresses from local cache servers in the network.

22. The cache server of claim 19, wherein the prefetching means includes

means for prefetching the MFU domain names and the associated addresses on a predetermined basis.

23. A cache server for supporting error functions of a local cache server of a network that processes domain name requests, the cache server comprising:

means for collecting a plurality of valid domain names;

means for generating validity codes corresponding to the plurality of valid domain names; and

means for transmitting to the local cache server the validity codes.

24. The cache server of claim 23, wherein the generating means includes

means for generating the validity codes using a hash coding.

25. A local cache server, connected to a local computer of a network, for processing domain name requests, the local cache server comprising:

a memory storing prefetched most frequently used (MFU) domain names and associated domain addresses;

means for receiving a request from the local computer for a domain address corresponding to a domain name;

means for determining whether the received domain name matches any of the prefetched MFU domain names; and

means for transmitting to the local computer the domain address associated with the received domain name.

26. The local cache server of claim 25, further including

means for determining the validity of the received domain name.

27. The local cache server of claim 25, further including

a memory for storing validity codes for valid domain names,

means for generating, in response to a determination that the received domain name does not match any of the prefetched MFU domain names, a check code of the received domain name, and

means for determining whether the check code matches any of the validity codes.

28. The local cache server of claim 25, further including

means for transmitting to the local computer, in response to a determination that the check code matches one of the validity codes, a signal indicating that the received domain name is valid.

29. The local cache server of claim 25, further including

means for transmitting to the local computer, in response to a determination that the check code does not match any of the validity codes, a signal indicating that the received domain name is invalid.

30. The local cache server of claim 25, further including

means for receiving the prefetched MFU domain names and the associated domain addresses from a cache server.

31. An article of manufacture capable of configuring a local cache server to process domain name requests, the article comprising program code to cause the local cache server to perform the steps of:

storing in a memory prefetched most frequently used (MFU) domain names and associated domain addresses;

receiving a request from the local computer for a domain address corresponding to a domain name;

determining whether the received domain name matches any of the prefetched MFU domain names; and

transmitting to the local computer the domain address associated with the received domain name.

32. An article of manufacture capable of configuring a cache server to efficiently process domain name requests, the article comprising program code to cause the cache server to perform the steps of:

prefetching most frequently used (MFU) domain names and associated domain addresses; and

transmitting to the local cache server the prefetched MFU domain names and the associated domain addresses, wherein the local cache server uses the prefetched MFU domain names and the associated domain addresses to process domain name requests.

FIG. 1

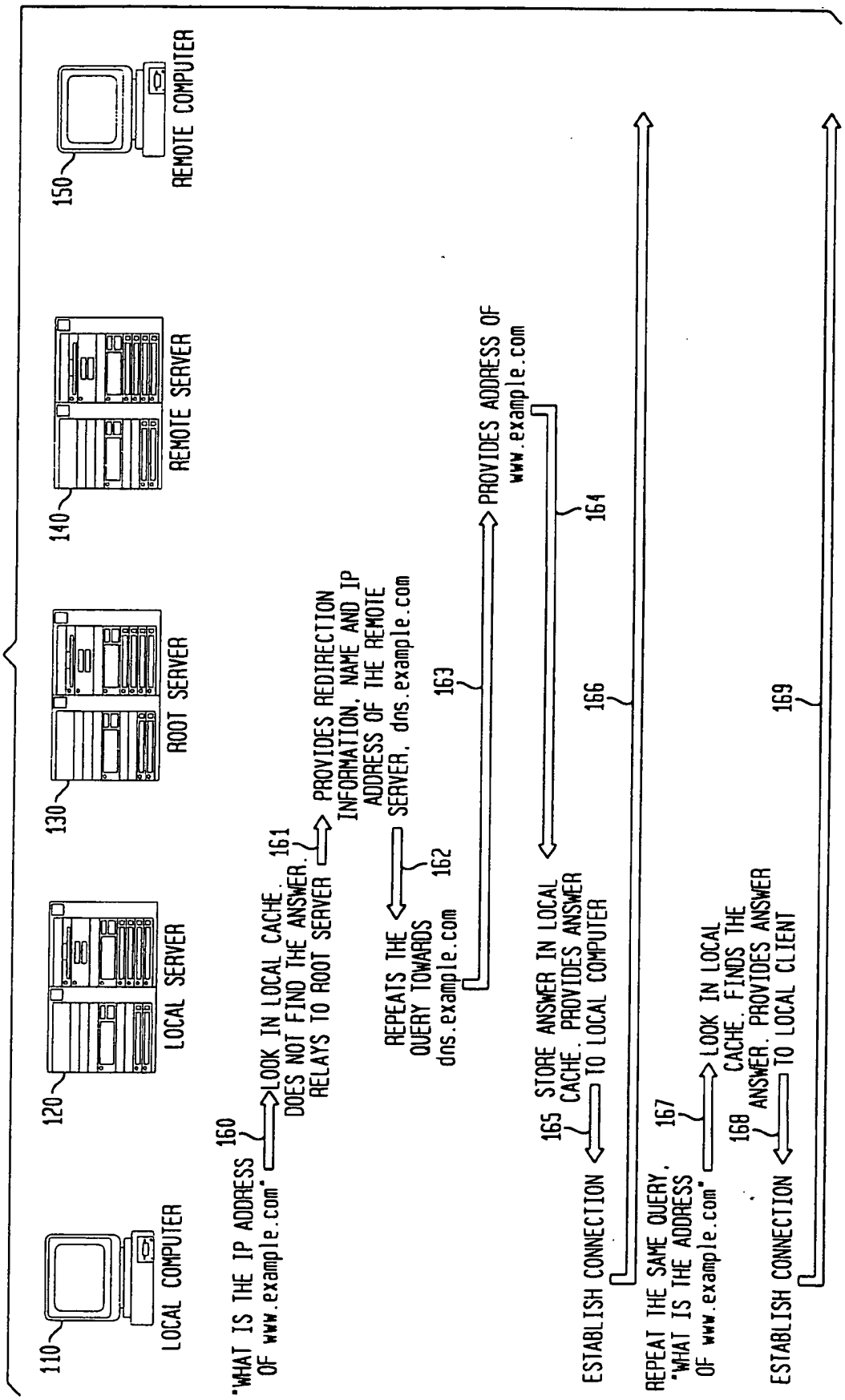

110
LOCAL COMPUTER
120
LOCAL SERVER
130
ROOT SERVER
140
REMOTE SERVER
150
REMOTE COMPUTER
"WHAT IS THE IP ADDRESS OF www.example.com"
160
LOOK IN LOCAL CACHE. DOES NOT FIND THE ANSWER. RELAYS TO ROOT SERVER
161
PROVIDES REDIRECTION INFORMATION, NAME AND IP ADDRESS OF THE REMOTE SERVER, dns.example.com
162
REPEATS THE QUERY TOWARDS dns.example.com
163
PROVIDES ADDRESS OF www.example.com
164
STORE ANSWER IN LOCAL CACHE. PROVIDES ANSWER TO LOCAL COMPUTER
165
ESTABLISH CONNECTION
166
REPEAT THE SAME QUERY, "WHAT IS THE ADDRESS OF www.example.com"
167
LOOK IN LOCAL CACHE. FINDS THE ANSWER. PROVIDES ANSWER TO LOCAL CLIENT
168
ESTABLISH CONNECTION
169

FIG. 2

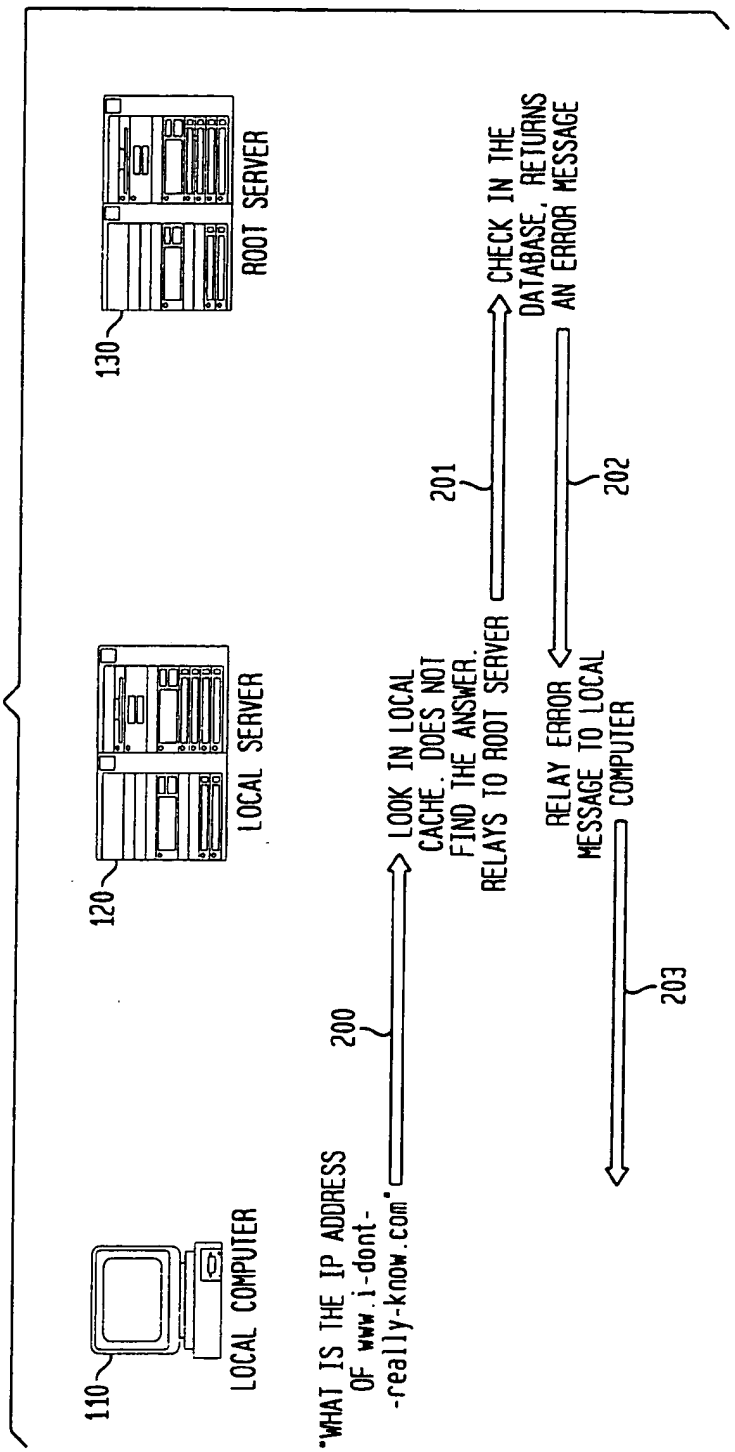

ROOT SERVER
130
LOCAL SERVER
120
LOCAL COMPUTER
110
"WHAT IS THE IP ADDRESS OF www.i-dont--really-know.com"
200
LOOK IN LOCAL CACHE. DOES NOT FIND THE ANSWER. RELAYS TO ROOT SERVER
201
CHECK IN THE DATABASE, RETURNS AN ERROR MESSAGE
202
RELAY ERROR MESSAGE TO LOCAL COMPUTER
203

FIG. 3

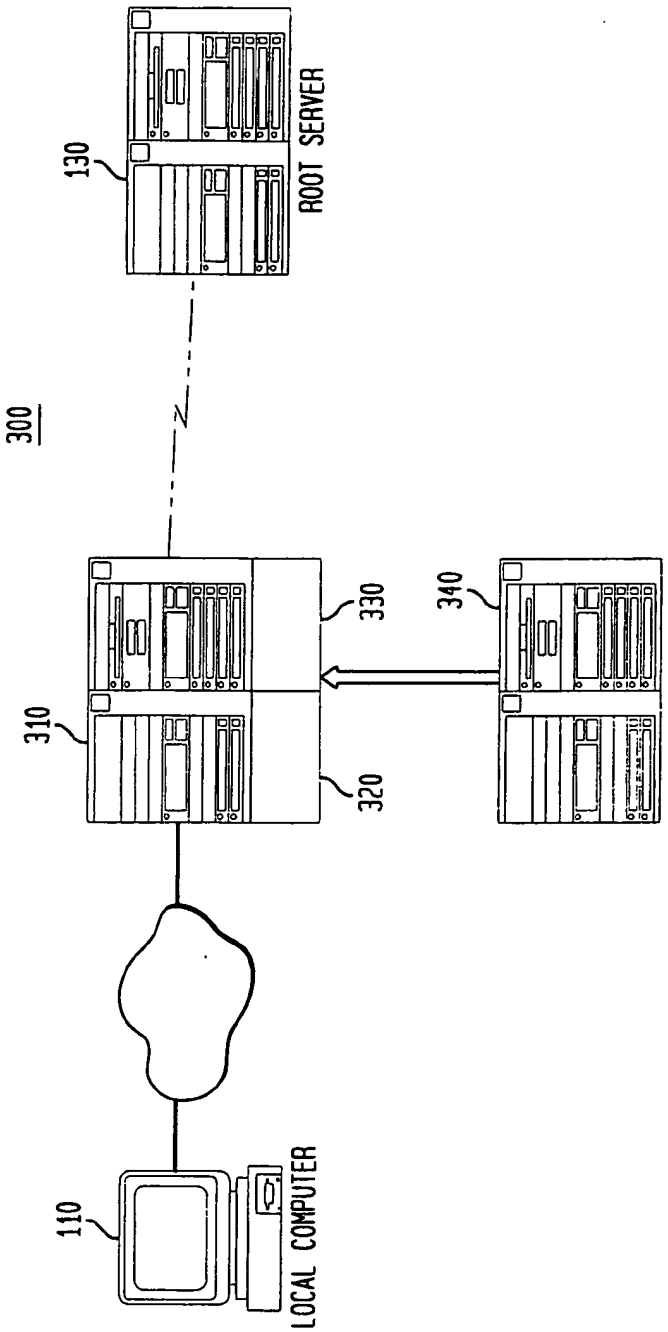


FIG. 4

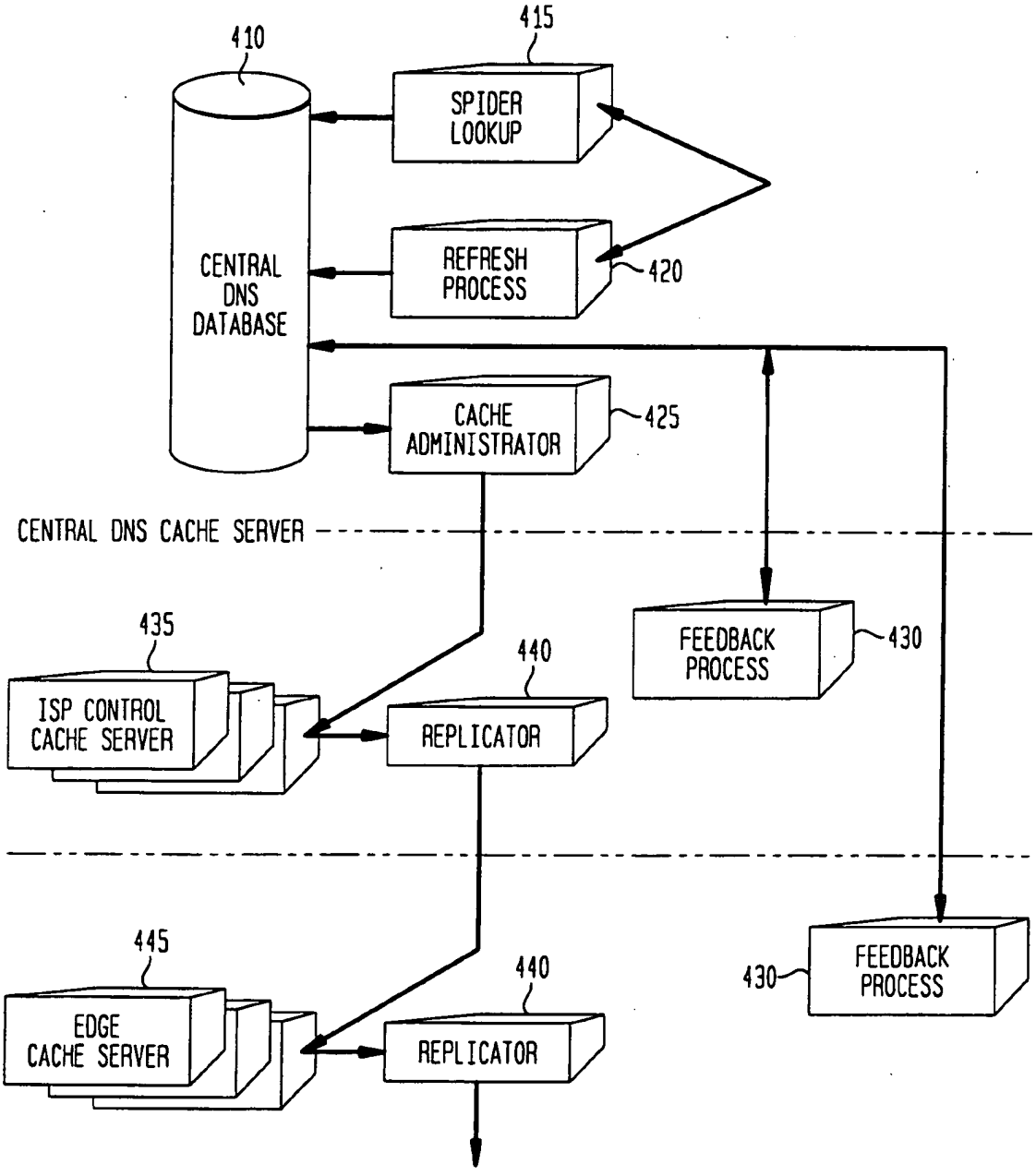

410
415
SPIDER LOOKUP
CENTRAL DNS DATABASE
REFRESH PROCESS
420
CACHE ADMINISTRATOR
425
CENTRAL DNS CACHE SERVER
435
ISP CONTROL CACHE SERVER
440
REPLICATOR
FEEDBACK PROCESS
430
445
EDGE CACHE SERVER
440
REPLICATOR
430
FEEDBACK PROCESS

FIG. 5

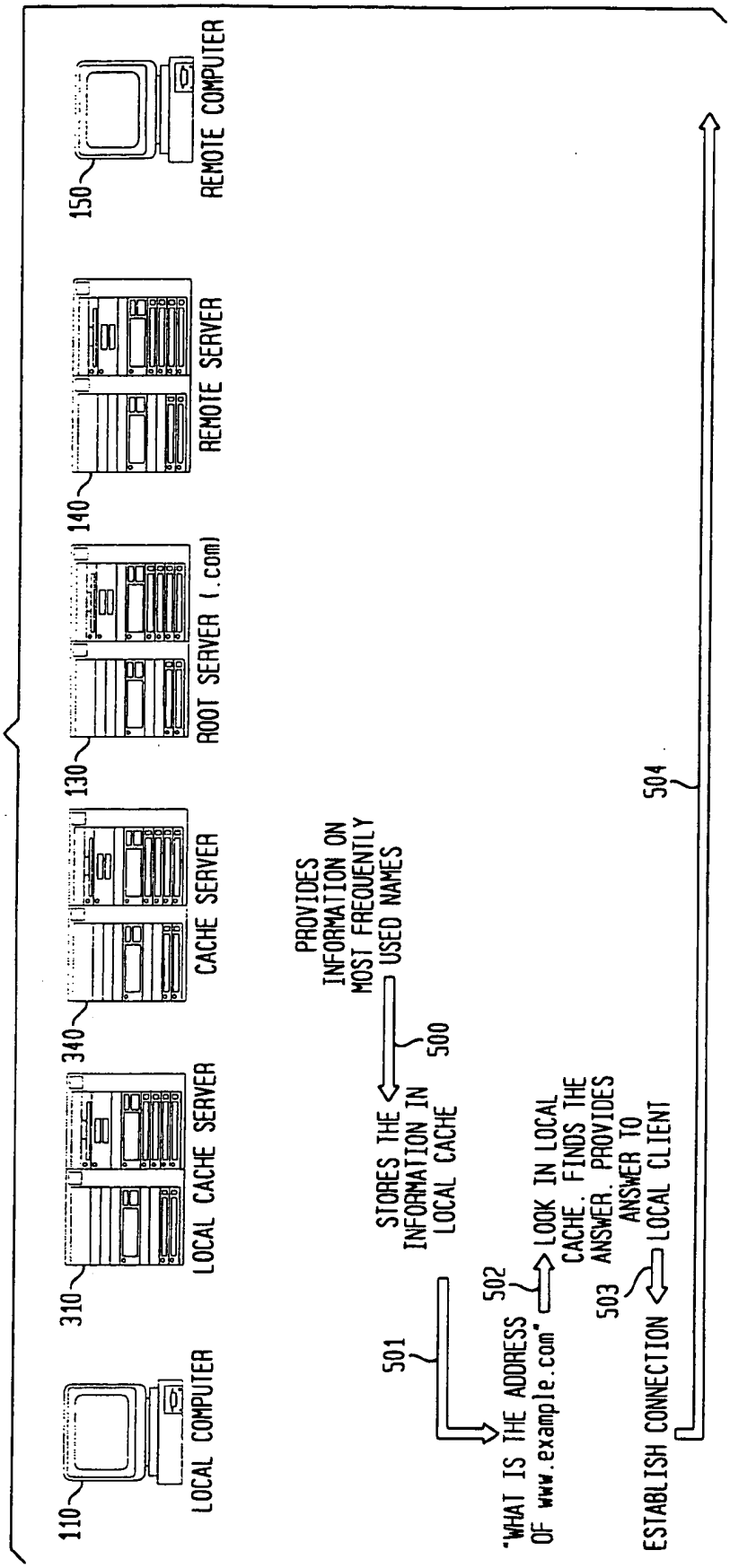
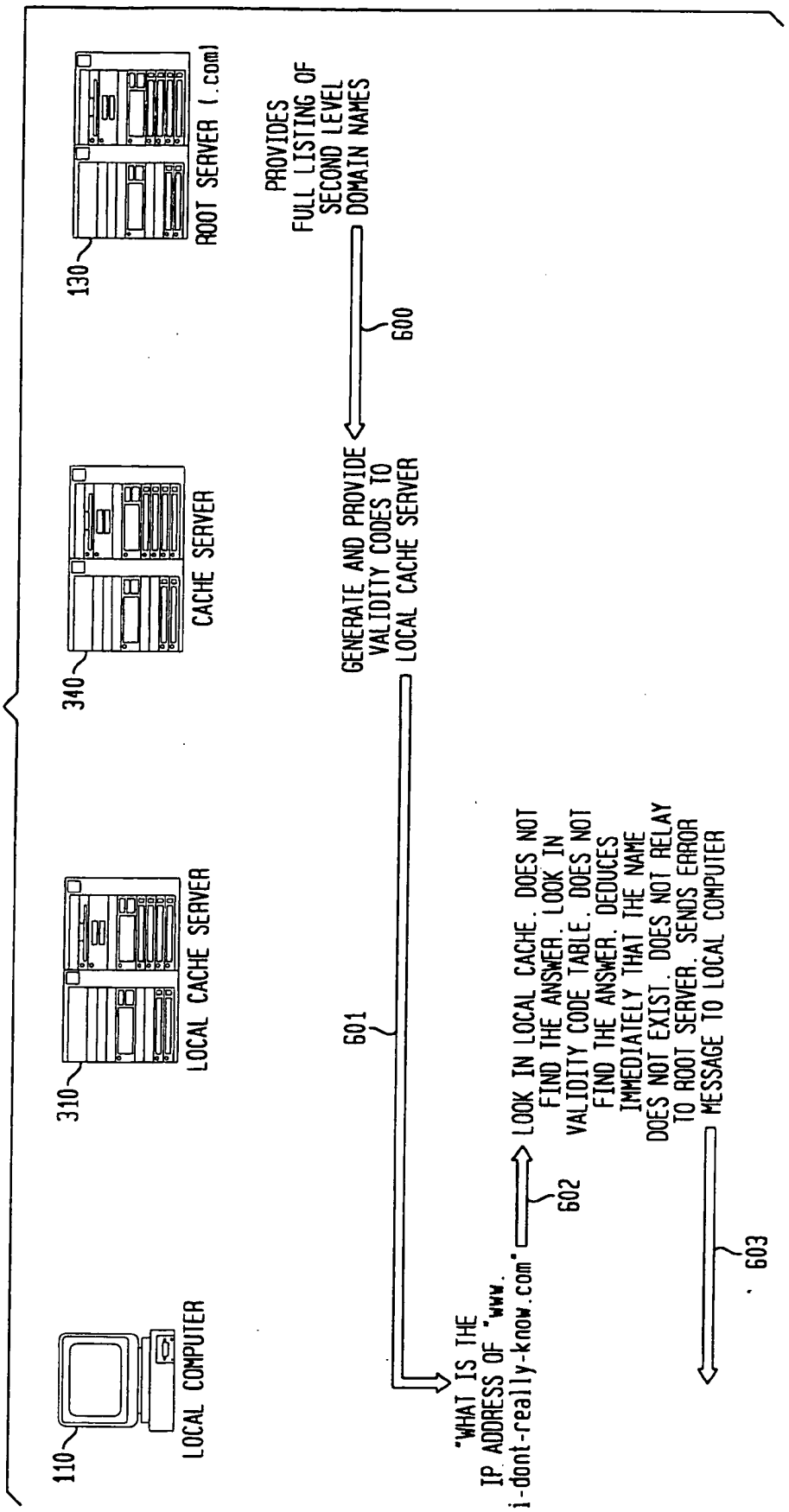

110
LOCAL COMPUTER
310
LOCAL CACHE SERVER
340
CACHE SERVER
130
ROOT SERVER (.com)
140
REMOTE SERVER
150
REMOTE COMPUTER
PROVIDES INFORMATION ON MOST FREQUENTLY USED NAMES
500
STORES THE INFORMATION IN LOCAL CACHE
501
"WHAT IS THE ADDRESS OF www.example.com"
502
LOOK IN LOCAL CACHE. FINDS THE ANSWER. PROVIDES ANSWER TO LOCAL CLIENT
503
ESTABLISH CONNECTION
504

FIG. 6


ROOT SERVER (.com)
130
PROVIDES FULL LISTING OF SECOND LEVEL DOMAIN NAMES
600
CACHE SERVER
340
GENERATE AND PROVIDE VALIDITY CODES TO LOCAL CACHE SERVER
LOCAL CACHE SERVER
310
601
"WHAT IS THE IP ADDRESS OF "www.i-dont-really-know.com"
LOOK IN LOCAL CACHE. DOES NOT FIND THE ANSWER. LOOK IN VALIDITY CODE TABLE. DOES NOT FIND THE ANSWER. DEDUCES IMMEDIATELY THAT THE NAME DOES NOT EXIST. DOES NOT RELAY TO ROOT SERVER. SENDS ERROR MESSAGE TO LOCAL COMPUTER
602
603
LOCAL COMPUTER
110

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/21239

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :Please See Extra Sheet.
US CL :Please See Extra Sheet.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : Please See Extra Sheet.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, Internet
domain name, cache, prefetch, web spider, hash, code, valid codes, check codes, hash codes

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	RFC 1035, www.dns.be/rfc/rfc1035.html, 13 February 1996 figs. on page 2, 4; sub-chapter 2.2	1 , 3 , 4 , 6 - 16,18,19,21-32
Y	www.cs.unc.edu/Courses/wwwc.public/ladd/search.html; 14 February 1995	2,5,17,20
Y	info.webcrawler.com/mak/projects/robots/faq.html; 1995	2,5,17,20
X	The Art of Computer Programming, Vol. 3 (KNUTH) 1981 sub-chapter 6.4 HASHING	8 , 9 , 1 2 - 14,23,24,27-29

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
19 JANUARY 1999	15 MAR 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 308-5357	Authorized officer GLENN BURGESS James R. Matthews Telephone No. (703) 305-4792

Form PCT/ISA/210 (second she

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/21239

A. CLASSIFICATION OF SUBJECT MATTER:
IPC (6):

H04L 12/00, 12/26
G06F 13/00, 15/00, 3/00, 9/06, 17/30
H04M 3/42
H04B 1/00

A. CLASSIFICATION OF SUBJECT MATTER:
US CL :

395/200.75, 200.33, 200.48, 200.49, 200.79, 615
370/254
379/219
340/825.52

B. FIELDS SEARCHED
Minimum documentation searched
Classification System: U.S.

395/200.75, 200.33, 200.48, 200.49, 200.79, 615
370/254
379/219
340/825.52